**What is Spring Cloud?  
A:** Spring Cloud Stream App Starters are Spring Boot based Spring Integration applications that provide integration with external systems. Spring Cloud Task. A short-lived microservices framework to quickly build applications that perform finite amounts of data processing.  
[Spring Cloud](https://www.javainuse.com/spring/springcloud)   
  
**Q: What are the advantages of using Spring Cloud?  
A:** When developing distributed microservices with Spring Boot we face the following issues-

* **Complexity associated with distributed systems-**  
  This overhead includes network issues, Latency overhead, Bandwidth issues, security issues.
* **Service Discovery-**  
  Service discovery tools manage how processes and services in a cluster can find and talk to one another. It involves a directory of services, registering services in that directory, and then being able to lookup and connect to services in that directory.
* **Redundancy-**  
  Redundancy issues in distributed systems.
* **Loadbalancing-**  
  Load balancing improves the distribution of workloads across multiple computing resources, such as computers, a computer cluster, network links, central processing units, or disk drives.
* **Performance issues-**  
  Performance issues due to various operational overheads.
* **Deployment complexities-**  
  Requirement of Devops skills.

**Q: What does one mean by Service Registration and Discovery? How is it implemented in Spring Cloud?  
A:** When we start a project, we usally have all the configurations in the properties file. As more and more services are developed and deployed, adding and modifying these properties become more complex. Some services might go down, while some the location might change. This manual changing of properties may create issues.  
Eureka Service Registration and Discovery helps in such scenarios. As all services are registered to the Eureka server and lookup done by calling the Eureka Server, any change in service locations need not be handled and is taken care of  
[Microservice Registration and Discovery with Spring cloud using Netflix Eureka.](https://www.javainuse.com/spring/spring_eurekaregister)   
  
**Q: What does one mean by Load Balancing ? How is it implemented in Spring Cloud?  
A:** In computing, load balancing improves the distribution of workloads across multiple computing resources, such as computers, a computer cluster, network links, central processing units, or disk drives. Load balancing aims to optimize resource use, maximize throughput, minimize response time, and avoid overload of any single resource. Using multiple components with load balancing instead of a single component may increase reliability and availability through redundancy. Load balancing usually involves dedicated software or hardware, such as a multilayer switch or a Domain Name System server process.  
In SpringCloud this can be implemented using Netflix Ribbon.  
[Spring Cloud- Netflix Eureka + Ribbon Simple Example](https://www.javainuse.com/spring/spring_ribbon)

**Q: What is Hystrix? How does it implement Fault Tolerance?  
A:** In computing, load balancing improves the distribution of workloads across multiple computing resources, such as computers, a computer cluster, network links, central processing units, or disk drives. Load balancing aims to optimize resource use, maximize throughput, minimize response time, and avoid overload of any single resource. Using multiple components with load balancing instead of a single component may increase reliability and availability through redundancy. Load balancing usually involves dedicated software or hardware, such as a multilayer switch or a Domain Name System server process.  
In SpringCloud this can be implemented using Netflix Ribbon.  
[Spring Cloud- Netflix Eureka + Ribbon Simple Example](https://www.javainuse.com/spring/spring_ribbon)   
  
**Q: What is Hystrix? How does it implement Fault Tolerance?  
A:** **Hystrix is a latency and fault tolerance library designed to isolate points of access to remote systems, services and 3rd party libraries, stop cascading failure and enable resilience in complex distributed systems where failure is inevitable.**  
Usually for systems developed using Microservices architecture, there are many microservices involved. These microservices collaborate with each other.  
Consider the following microservices-

Suppose if the microservice 9 in the above diagram failed, then using the traditional approach we will propagate an exception. But this will still cause the whole system to crash anyways.  
This problem gets more complex as the number of microservices increase. The number of microservices can be as high as 1000. This is where hystrix comes into picture  
We will be the Fallback method feature of Hystrix for this scenario. We have two services employee-consumer consuming the service exposed by the employee-producer.  
The simplified diagram is as below-

Now suppose due to some reason the employee-producer exposed service throws an exception. In this case using Hystrix we define a fallback method. This fallback method should have the same return type as the exposed service. In case of exception in the exposed service the fallback method will return some value. [Spring Cloud- Netflix Hystrix Fallback method Simple Example](https://www.javainuse.com/spring/spring_hystrix)

**Q: What is Hystrix Circuit Breaker? Need for it?  
A:** Due to some reason the employee-producer exposed service throws an exception. In this case using Hystrix we defined a fallback method. In case of exception in the exposed service the fallback method returned some default value.

If the exceptions keep on occuring in the firstPage method() then the Hystrix circuit will break and the employee consumer will skip the firtsPage method all together and directly call the fallback method.

The purpose of circuit breaker is to give time to the first page method or other methods that the firstpage method might be calling and is causing the exception to recover. It might happen that on less load the issue causing the exceptions have better chance of recovering

[Spring Cloud- Circuit Breaker using Netflix Hystrix Simple Example](https://www.javainuse.com/spring/spring_hystrix_circuitbreaker)   
  
**Q: What is Netflix Feign? What are its advantages?  
A:** Feign is a java to http client binder inspired by Retrofit, JAXRS-2.0, and WebSocket. Feign's first goal was reducing the complexity of binding Denominator uniformly to http apis regardless of restfulness. Previous examples in the employee-consumer we consumed the REST services exposed by the employee-producer using **REST Template**

But we had to write a lot of code to perform following-

* For Load balancing using Ribbon.
* Getting the Service instance and then the Base URL.
* Make use of the REST Template for consuming service.

The previous code was as below

@Controller

public class ConsumerControllerClient {

**@Autowired**

**private LoadBalancerClient loadBalancer;**

**public void getEmployee() throws RestClientException, IOException {**

**ServiceInstance serviceInstance=loadBalancer.choose("employee-producer");**

**System.out.println(serviceInstance.getUri());**

**String baseUrl=serviceInstance.getUri().toString();**

baseUrl=baseUrl+"/employee";

RestTemplate restTemplate = new RestTemplate();

ResponseEntity<String> response=null;

try{

response=restTemplate.exchange(baseUrl,

HttpMethod.GET, getHeaders(),String.class);

}catch (Exception ex)

{

System.out.println(ex);

}

System.out.println(response.getBody());

}

The previous code, there are chances of exceptions like NullPointer and is not optimal. We will see how the call is made much easier and cleaner using Netflix Feign. If the Netflix Ribbon dependency is also in the classpath, then Feign also takes care of load balancing by default.  
[Spring Cloud- Netflix Feign Simple Example](https://www.javainuse.com/spring/spring-cloud-netflix-feign-tutorial)   
  
**Q: What is Spring Cloud Bus? Need for it?  
A:** Consider the scenario that we have multiple applications reading the properties using the Spring Cloud Config and the Spring Cloud Config in turn reads these properties from GIT.  
Consider the below example where multiple employee producer modules are getting the property for Eureka Registration from Employee Config Module.

What will happen if suppose the eureka registration property in GIT changes to point to another Eureka server. In such a scenario we will have to restart the services to get the updated properties. There is another way of using Actuator Endpoint **/refresh**. But we will have to individually call this url for each of the modules. For example if Employee Producer1 is deployed on port 8080 then call **http://localhost:8080/refresh**. Similarly for Employee Producer2 **http://localhost:8081/refresh** and so on. This is again cumbersome. This is where Spring Cloud Bus comes into picture.

The Spring Cloud Bus provides feature to refresh configurations across multiple instances. So in above example if we refresh for Employee Producer1, then it will automatically refresh for all other required modules. This is particularly useful if we are having multiple microservice up and running. This is achieved by connecting all microservices to a single message broker. Whenever an instance is refreshed, this event is subscribed to all the microservices listening to this broker and they also get refreshed. The refresh to any single instance can be made by using the endpoint **/bus/refresh**  
[Spring Cloud Tutorial - Publish Events Using Spring Cloud Bus](https://www.javainuse.com/spring/cloud-stream-bus)

What is Spring Cloud ?

Spring Cloud Stream App Starters are Spring Boot based Spring Integration applications that provide integration with external systems. Spring Cloud Task. A short-lived microservices framework to quickly build applications that perform finite amounts of data processing

What is microservice ?

It is loosely coupled , distributed small service that allows to decompose large application into small , well defined manageable components.so that they are completely independent of one another

Mention few characteristics of microservice ?

* It breaks down logic into small well defined component.
* Each component has a small domain of responsibility.It is deployed completely independent of another microservice.
* Microservice communicate or exchange data using lightweight protocols like HTTP and JSON.
* As Microservice is small and independent in nature so small team can work on well-defined areas of responsibility
* Microservice owns their data structure and data source.Database that keeps the microservice's data, can be locked down to only allow that service to access it

What is Netflix Feign ?

Feign is a java to http client binder inspired by Retrofit.

Mention few Microservice patterns ?

* Routing Pattern
* Core Pattern
* Security Pattern
* Build and deployment pattern
* Client resiliency pattern
* Logging and tracing patterns

What is Microservice Routing Pattern ?

In Routing Pattern ,physical address of services are kept abstract and only single point of entry for service call is provided.In Routing Pattern whenever a client application wants to consume any service then it discovers the location of service and then is routed to that service.

What is spring cloud config ?

Microservices have a lot of configuration specially related to environment. Spring cloud config handles the management of application configuration data through a centralized service.By this way all microservivces have same configuration. and configuration will be separate from microservivces.It has its own repository.

Can Spring cloud config integrates with open sources for repository like git ?

Yes

What is the use of spring cloud stream ?

Spring cloud stream allows to integrate microservices with message brokers

What is Spring cloud sleuth ?

By this we can integrate unique tracking identifiers into the HTTP calls and message channels.That helps to track a transaction as it flows across the different services in your application

What is Spring cloud security ?

It is an authentication and authorization framework that can control who can access our services and what they can do with your services.In spring cloud security microservices communicate using token.Receiving service first check token to validate caller.

Give some example where microservices are not powerful ?

* Microservices instance are deployed in separate servers.If an application has many Microservices then we need to maintain many server and complexity will be higher.
* If application is small then building microservices for distributed model is very complex.

Why cloud based application configuration should be centralized ?

In cloud based application , it's possible that many microservices are deployed and its really difficult to maintain separate repository for each microservice. So its good to maintain centralize repository to hold application configuration that reduces number of repository.

In which profile Spring cloud configuration server will run if filesystem is used to store configuration information ?

native

What is the annotation to make enable spring boot application for spring cloud config server ?

@EnableConfigServer

Give one example of Spring cloud config server using file system ?

[Spring Cloud Config Example.](http://preparationforinterview.com/preparationforinterview/spring-cloud-config)

Which property we need to set to make decryption disable at server ?

spring.cloud.config.server.encrypt.enabled: false

Give one example where spring cloud encrypt and decrypt properties ?

[Spring Cloud Config Example with encryption.](http://preparationforinterview.com/preparationforinterview/spring-cloud-config-encryption)

What is service discovery ?

In a distributed architecture , we used to find the physical address of the service where that service is deployed.This concept is called service discovery.

Why service discovery in microservice is critical ?

* As service consumer does not know the physical address of the service instance so using service discovery application developer can horizontally scale up and down service instance running in an environment.
* It helps to increase application resiliency because service discovery removes unavailable service instance from its list.

Mention few benefits for service discovery mechanism?

* Availability : Service lookups is shared among all nodes of service discovery cluster.So even a node becomes unavailable then others node take over.
* Sharing Instances : each node in cluster shares instances of services.
* Fault tolerant : If any service instance is not healthy then service discovery removes it from its table.
* Load balanced : Service discovery ensures that when when service invocation happens then invocation is spread across all instances.

What is client side load balancing in service discovery mechanism ?

When a service consumer wants to consume any service , he needs service location. To get service location he calls to service discovery and gets all service instances of the service which consumers wants to consume and cache it locally.

Next time again if he calls same service then he gets from local cache.We say load balancing because it uses algorithm so that every instance gets equal chance.

What annotation is required to make a service as Eureka service ?

@EnableEurekaServer

What is the significance of **eureka.instance.preferIpAddress** entry at service 's properties file while doing registery at Eureka service ?

By default Eureka assumes that service will be registered as hostname but by making this entry true we can inform to Eureka that service should be registered as IP address.

Mention few client library by which service consumer can interact with Ribbon ?

* Spring Discovery client
* Ribbon Aware Spring RestTemplate
* Netflix Feign client

Give one example of client library by which service consumer can interact with Ribbon ?

[Netflix Eureka Service Discovery Client](http://preparationforinterview.com/preparationforinterview/microservice-registration-service-discovery)

What is the use of client side resiliency pattern ?

It protects the client from crashing when remote resource is failing due to poor performance or error.It makes client fail fast.

Mention few client resiliency patterns ?

* Fallback
* Bulkhead
* Circuit breakers
* Client-side load balancing

What is Client-side load balancing ?

Client-side load balancing caches microservices endpoints retrieved from service discovery.

What is Fallback ?

Fallback executes any alternative flow if it exists in case remote service 's call is failing.

What is Circuit breakers ?

It stops client to call remote service if remote service is repeatedly failing.

What is Bulkheads ?

It segregates different service call on service client so that poor service does not use all resources on client.

What is service gateway ?

In distributed microservices application , cross cutting concern like logging , security are done through separate independent service and call to other service are routed through this independent service.This service is called service gateway.It acts as an intermediary between the service client and a service being invoked.

What is the use of ignored-services attribute in zuul service gateway?

By default when we do manually route mapping for service registered with Eureka then zuul exposes automatic route mapping as well.If we want to expose only manually route mapping then ignored-services attribute is used and eureka id based routes are excluded.

What type of filters are supported by Zuul ?

* Pre filters
* Post filters
* Route filters